

Let's Teach Architecting High Quality Software

Linda Northrop

CSEET 2006

© 2006 Carnegie Mellon University

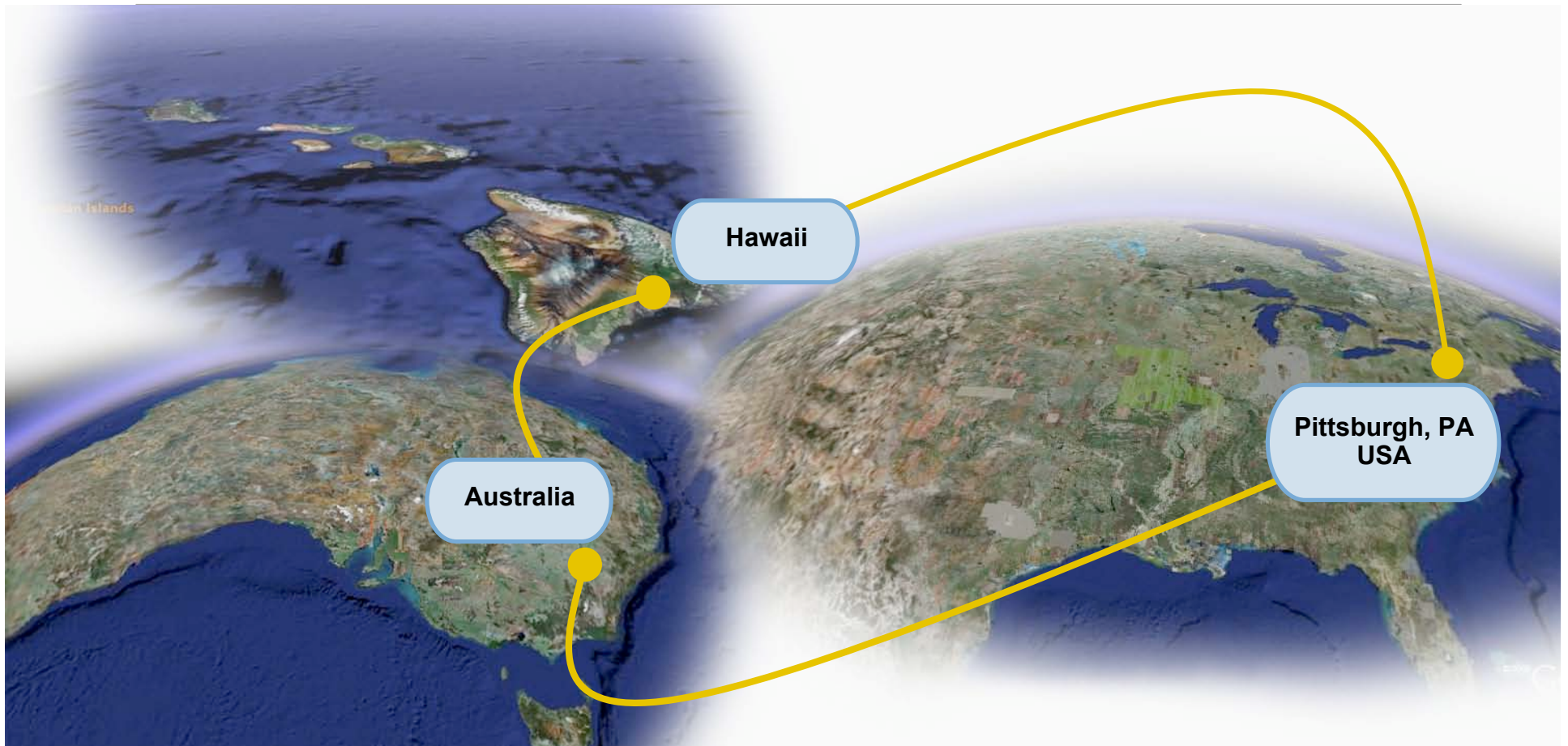


Software Engineering Institute

| CarnegieMellon

Report Documentation Page				Form Approved OMB No. 0704-0188	
Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.					
1. REPORT DATE APR 2006		2. REPORT TYPE		3. DATES COVERED 00-00-2006 to 00-00-2006	
4. TITLE AND SUBTITLE Let's Teach Architecting High Quality Software				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Carnegie Mellon University ,Software Engineering Institute (SEI),Pittsburgh,PA,15213				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited					
13. SUPPLEMENTARY NOTES 19th Conference on Software Engineering Education and Training (CSEET 2006), April 19-21, Turtle Bay, HI.					
14. ABSTRACT					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT Same as Report (SAR)	18. NUMBER OF PAGES 73	19a. NAME OF RESPONSIBLE PERSON
a. REPORT unclassified	b. ABSTRACT unclassified	c. THIS PAGE unclassified			

MY GEOGRAPHY LESSON



Let's Teach Architecting High Quality Software Software

© 2006 Carnegie Mellon University



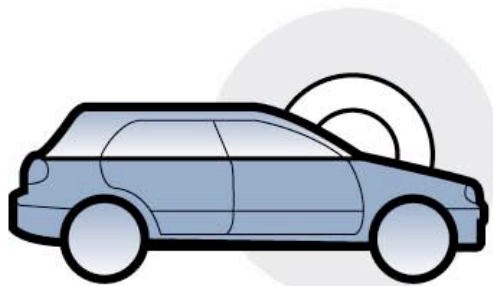
Software Engineering Institute

CarnegieMellon

Let's Teach Architecting High Quality Software

Page 3

SOFTWARE PERVADES OUR WORLD



Software is an integral part of our everyday lives.

Software has become the bottom line for many organizations who never envisioned themselves in the software business.

© 2006 Carnegie Mellon University



Software Engineering Institute

CarnegieMellon

Let's Teach Architecting High Quality Software

Page 4

much is required of software.

© 2006 Carnegie Mellon U

© 2006 Carnegie Mellon University

Let's Teach Architecting High Quality Software Quality

© 2006 Carnegie Mellon University



Software Engineering Institute

CarnegieMellon

Let's Teach Architecting High Quality Software

Page 6

QUALITY

Quality software is software that is fit for its intended purpose.



© 2006 Carnegie Mellon University



Let's Teach Architecting High Quality Software

High Quality

© 2006 Carnegie Mellon University



Software Engineering Institute

CarnegieMellon

Let's Teach Architecting High Quality Software

Page 8

HIGH QUALITY

High quality software meets business goals and user needs.
It has the right features and the right attributes.



© 2006 Carnegie Mellon University



UNIVERSAL BUSINESS GOALS

INCREASED MARKET SHARE
QUICK (OR RIGHT) TIME TO MARKET
EFFECTIVE USE OF LIMITED RESOURCES
PRODUCT ALIGNMENT
LOW-COST PRODUCTION
LOW-COST MAINTENANCE
MARKET AGILITY
MIND SHARE



A diagram illustrating the relationship between universal business goals and competitive advantage. On the left, a vertical list of eight goals is presented. A vertical line extends from the top of this list, with a horizontal line crossing it at the level of 'PRODUCT ALIGNMENT'. This horizontal line leads into a blue chevron-shaped box on the right, which contains the text 'COMPETITIVE ADVANTAGE'. The top and bottom of the vertical line are capped with dotted lines.

COMPETITIVE
ADVANTAGE

© 2006 Carnegie Mellon University



THE ULTIMATE UNIVERSAL GOAL



© 2000 Carnegie Mellon University



USER NEEDS

- Required capability
- Low learning threshold
- Ease of use
- Predictable behavior
- Dependable service
- Timely response
- Timely throughput
- Protection from unintended intruders and viruses
-

Software product goals should address user needs.

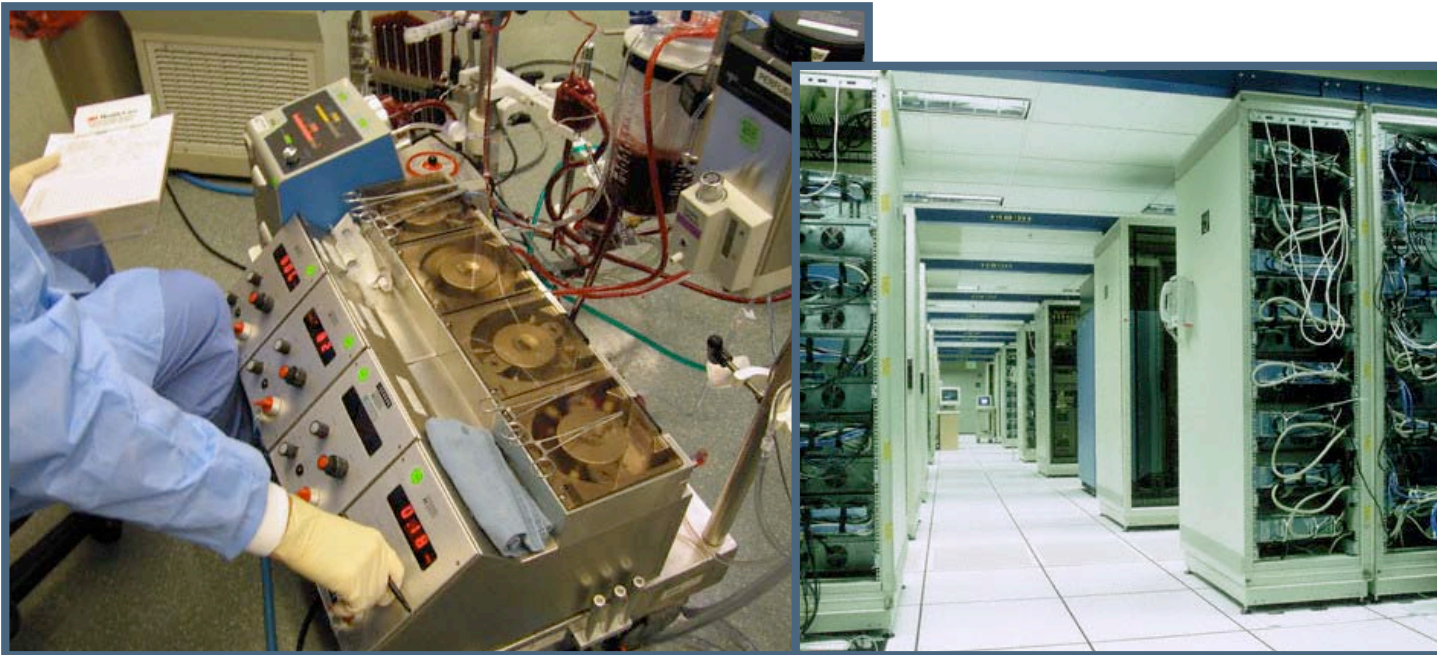
© 2006 Carnegie Mellon University



THE RIGHT FEATURES

SOFTWARE NEEDS TO HAVE THE RIGHT FUNCTIONALITY:

The software does what I want it to do and not (a lot) more.



© 2006 Carnegie Mellon University



THE RIGHT QUALITY ATTRIBUTES

Quality attributes include

- Performance
- Availability
- Usability
- Modifiability
- Security
- Etc.

Quality attribute requirements stem from business and product goals.

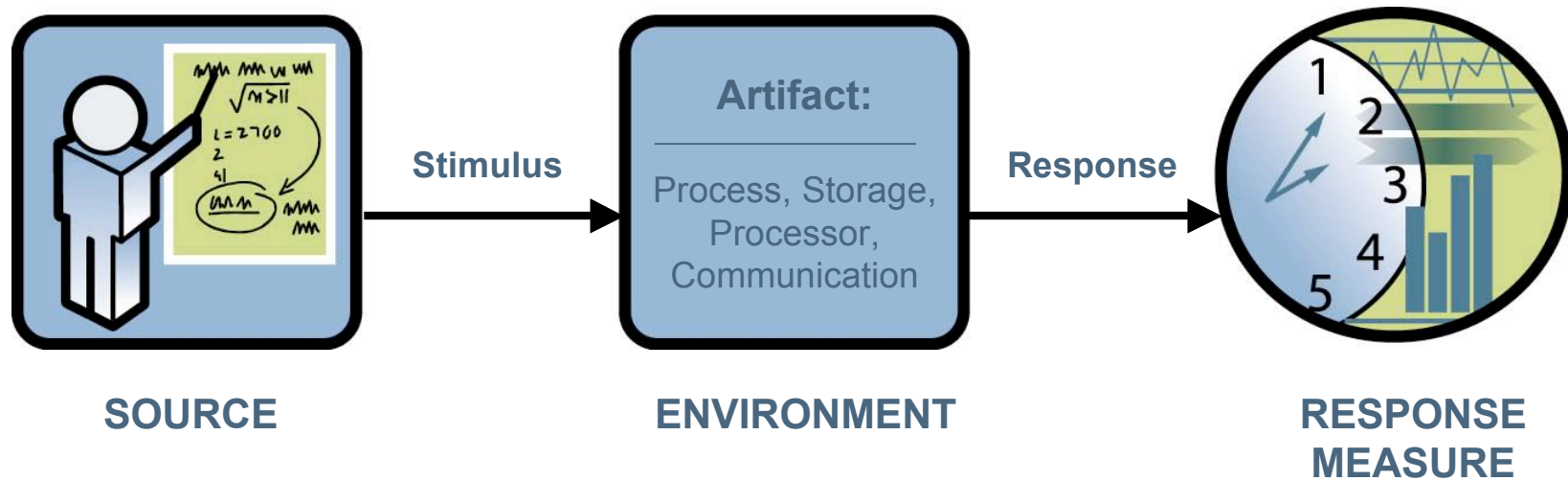
Key quality attributes need to be characterized in a system-specific way.

Scenarios are a powerful way to characterize quality attributes and represent stakeholder views.

© 2006 Carnegie Mellon University



PARTS OF A QUALITY ATTRIBUTE SCENARIO



© 2006 Carnegie Mellon University



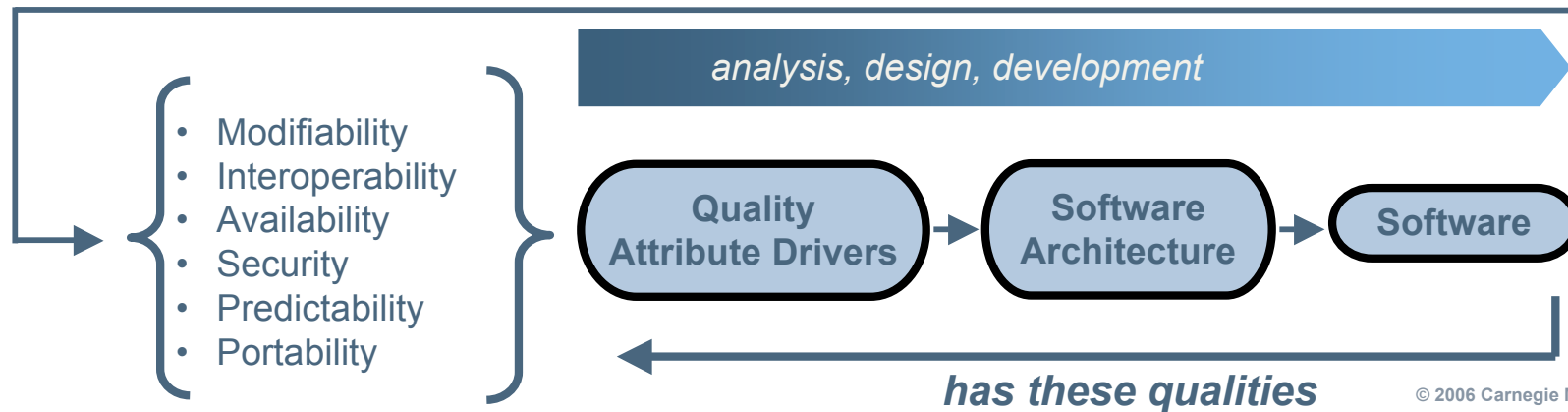
SOFTWARE SYSTEM DEVELOPMENT



Functional
Software
Requirements

If function were all that mattered, any monolithic software would do, ..**but other things matter...**

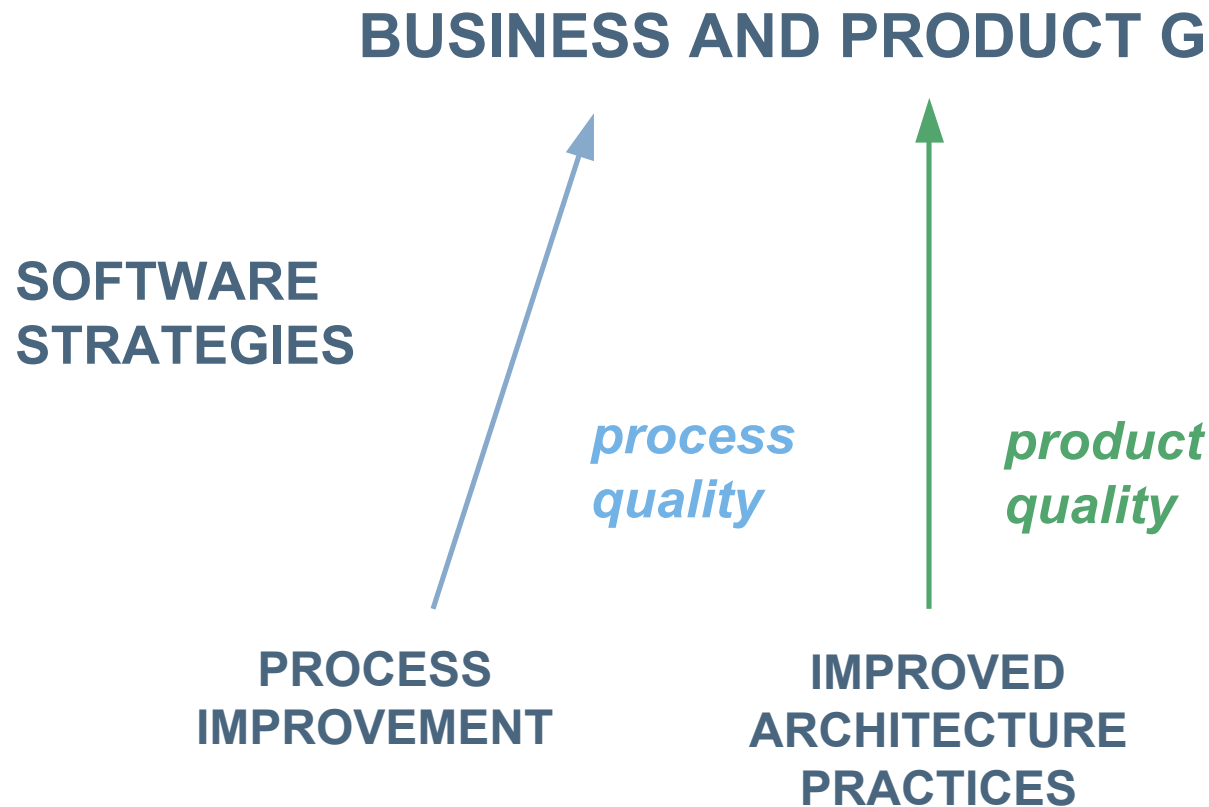
The important quality attributes and their characterizations are key.



© 2006 Carnegie Mellon University



SOFTWARE STRATEGIES ARE NEEDED



© 2006 Carnegie Mellon University



Let's Teach Architecting High Quality Software Architecture

© 2006 Carnegie Mellon University



Software Engineering Institute

CarnegieMellon

Let's Teach Architecting High Quality Software

Page 18

WHAT WE NEED IN SOFTWARE

Well-designed software architecture that

- lays out the basic elements of construction
- is known to satisfy important quality goals

Well-defined parts – components that

- have specified roles and interfaces
- have known properties
- behave predictably in a given assembly

Well-defined production plan that prescribes

- the order and method of assembly
- individual and team goals

© 2006 Carnegie Mellon University



WHAT WE NEED IN SOFTWARE

Well-designed software architecture that

- **lays out the basic elements of construction**
- **is known to satisfy important quality goals**

Well-defined parts – components that

- have specified roles and interfaces
- have known properties
- behave predictably in a given assembly

Well-defined production plan that prescribes

- the order and method of assembly
- individual and team goals

© 2006 Carnegie Mellon University



FOCUS: SOFTWARE ARCHITECTURE

From our experience

The quality and longevity of a software-intensive system is largely determined by its architecture.

Many large system and software failures point to

- inadequate software architecture education and practices
- the lack of any real software architecture evaluation early in the life cycle

Risk mitigation early in the life cycle is key.

- Mid-course correction is possible before great investment.
- Risks don't become problems that have to be addressed during integration and test.

© 2006 Carnegie Mellon University



WITHOUT SOFTWARE ARCHITECTURE FOCUS

Poorly designed software architectures result in

- greatly inflated integration and test costs
- inability to sustain systems in a timely and affordable way
- lack of system robustness
- in the worst case, product or project cancellation
- in all cases, failure to best support the user

© 2006 Carnegie Mellon University



WHY IS SOFTWARE ARCHITECTURE IMPORTANT?

Represents *earliest* design decisions



- hardest to change
- most critical to get right
- communication vehicle among stakeholders

First design artifact addressing



- performance
- modifiability
- reliability
- security

Key to systematic *reuse*



- transferable, reusable abstraction

The **right architecture** paves the way for system **success**.
The **wrong architecture** usually spells some form of **disaster**.

© 2006 Carnegie Mellon University



WHAT IS A SOFTWARE ARCHITECTURE?

Informally, software architecture is the blueprint describing system composition.

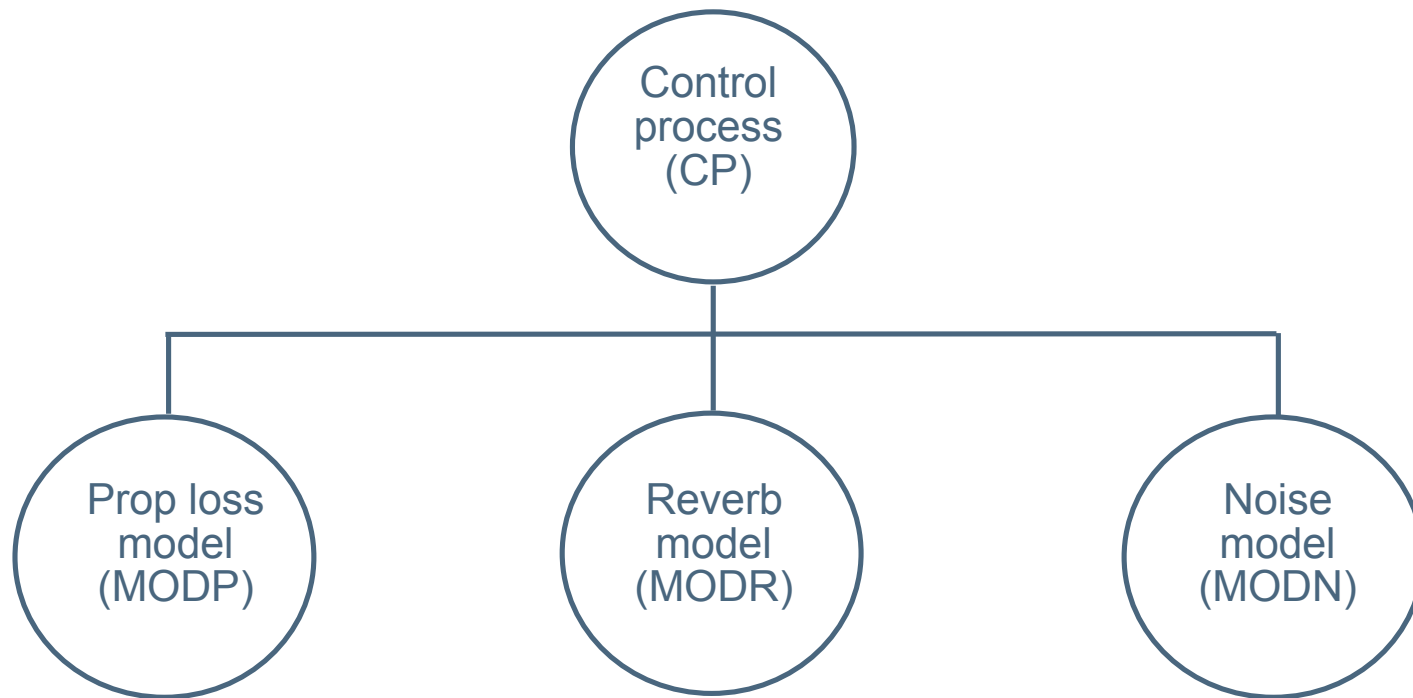
A software architecture is often depicted using an ad hoc box-and-line drawing of the system that is intended to solve the problems articulated by the specification.

- Boxes show elements or “parts” of the system.
- Lines show relationships among the parts.

© 2006 Carnegie Mellon University



A SOFTWARE ARCHITECTURE (?)



© 2006 Carnegie Mellon University



DEFINITION: SOFTWARE ARCHITECTURE

“The **software architecture** of a program or computing system is the **structure or structures** of the system, which comprise the **software elements**, the **externally visible properties** of those elements, and the **relationships among** them.”¹

¹ Bass, L.; Clements; P. & Kazman, R. *Software Architecture in Practice, Second Edition*. Boston, MA: Addison-Wesley, 2003.



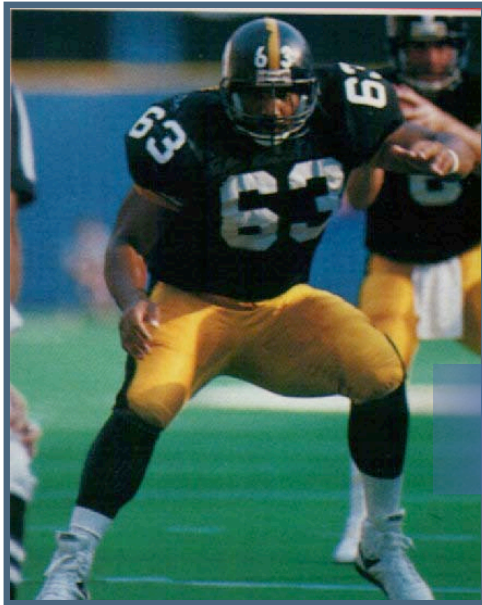
IMPLICATIONS OF OUR DEFINITION

- Software architecture is an abstraction of a system.
- Software architecture defines the properties of elements.
- Systems can and do have many structures.
- Every software-intensive system *has* an architecture.
- Just having an architecture is different from having an architecture that is known to everyone.
- If you don't develop an architecture, you will get one anyway – **and you might not like what you get!**

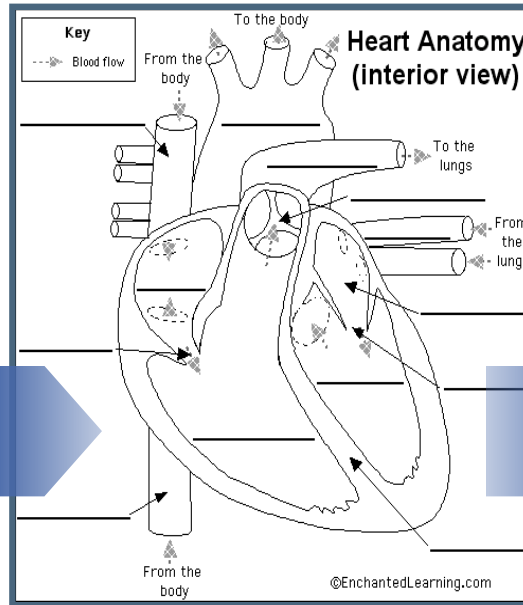
© 2006 Carnegie Mellon University



STRUCTURES AND VIEWS – 1



A human body comprises multiple *structures*.



a *static* view of one human *structure*



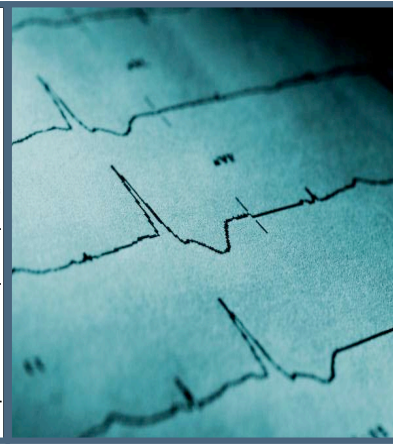
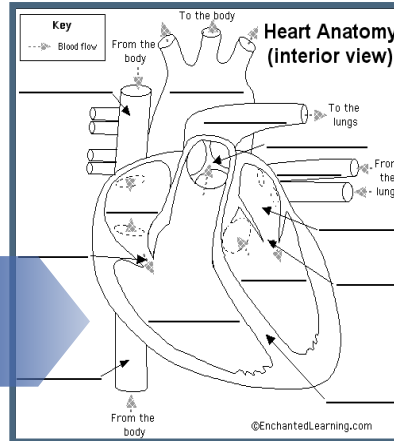
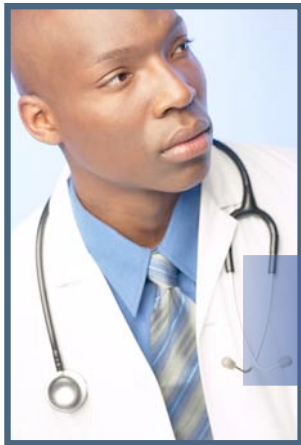
a *dynamic* view of that *structure*

*One body has many structures, and those structures have many views.
So it is with software...*

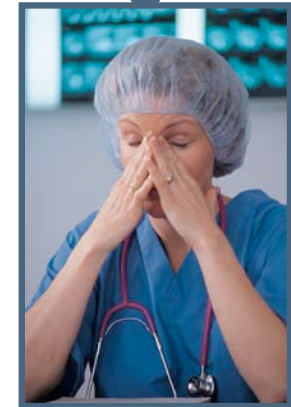
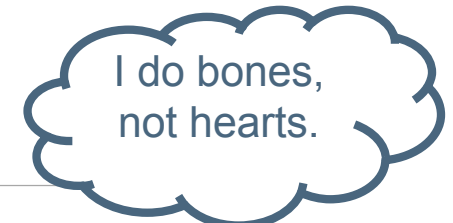
© 2006 Carnegie Mellon University



STRUCTURES AND VIEWS – 2



These views are needed by the cardiologist...



...but will they work for the orthopedist?

*Different stakeholders are interested in different structures.
Views must represent the structures in which the stakeholders are interested.
So it is with software...*

© 2006 Carnegie Mellon University



STRUCTURES AND VIEWS – 3

You should know

- **structure** – an actual set of architectural elements as they exist in software or hardware
- **view** – a representation of a coherent set of architectural elements, as written by and read by system stakeholders.
 - A view consists of a representation of a set of elements and the relations among them.

You should provide

- views that help evaluators and stakeholders understand the software architecture.

© 2006 Carnegie Mellon University



THIS IS WHAT HAPPENS



**FOR
SALE**

without careful architectural design.

And so it is with software.

© 2006 Carnegie Mellon University



Software Engineering Institute

CarnegieMellon

Let's Teach Architecting High Quality Software

Page 31

OTHER ARCHITECTURES - 1

Enterprise architectures are a means for describing business structures and processes that connect business structures.¹

- focus on business processes, dataflow, systems (including software packages), and their interconnection
- do not address the details of software design
- DoDAF, FEAR, and TEAF are generally regarded as enterprise architectures.

¹ Zachman, John A., "A Framework for Information Systems Architecture." *IBM Systems Journal*, 26, 3 (1987): 276-292.

© 2006 Carnegie Mellon University



OTHER ARCHITECTURES - 2

A **system architecture** is a means for describing the elements and interactions of a complete system including its hardware elements and its software elements.

System architecture is concerned with the elements of the system and their contribution toward the goal of the system, but not with their substructure.

System Architecture: *“The fundamental and unifying system structure defined in terms of system elements, interfaces, processes, constraints, and behaviors.”*¹

Systems Engineering is a design and management discipline useful in designing and building large, complex, and interdisciplinary systems.²

¹ Rechtin, E. *Systems Architecting: Creating and Building Complex Systems*. Englewood Cliffs, NJ : Prentice-Hall, 1991.

² International Council On Systems Engineering (INCOSE), Systems Architecture Working Group, 1996.



WHERE DOES SOFTWARE ARCHITECTURE FIT?

Enterprise architecture and system architecture provide an environment in which software lives.

- Both provide requirements and constraints to which software architecture must adhere.
- Elements of both are likely to contain software architecture.
- **Neither are a substitute for or obviate a software architecture.**

In large, complex, software-intensive systems **both software and system architectures are critical** for ensuring that the system is fit for the intended purpose.

© 2006 Carnegie Mellon University



Let's Teach Architecting High Quality Software

Architecting

© 2006 Carnegie Mellon University



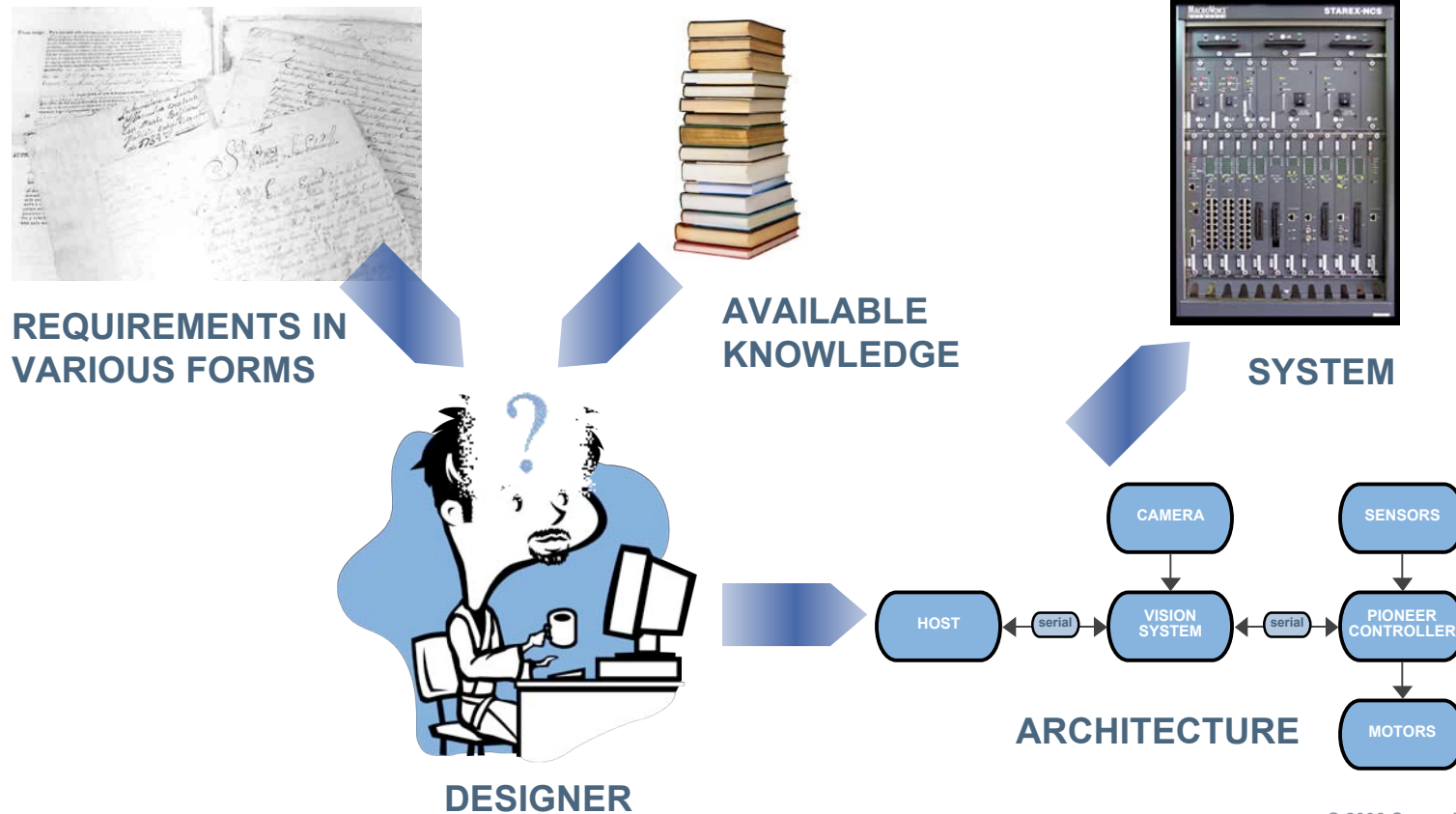
Software Engineering Institute

CarnegieMellon

Let's Teach Architecting High Quality Software

Page 35

REQUIREMENTS BEGET DESIGN



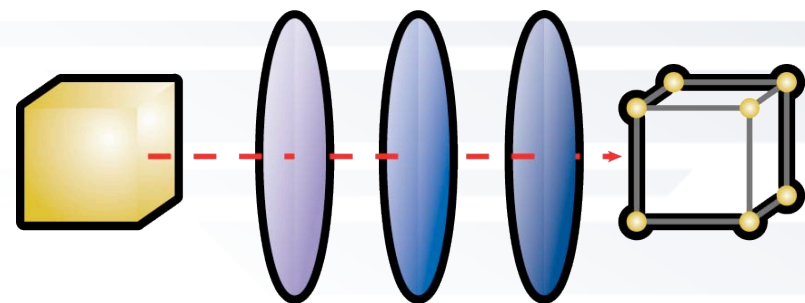
FACTORS INFLUENCING ARCHITECTURES

Where do architectures come from?

System requirements, constraints, business and product goals certainly, but that's not all.

Architectures are influenced by

- stakeholders of a system
- technical and organizational factors
- architect's background



© 2006 Carnegie Mellon University



INFLUENCE OF SYSTEM STAKEHOLDERS – 1

Stakeholders have an interest in the construction of a software system.

Stakeholders might include

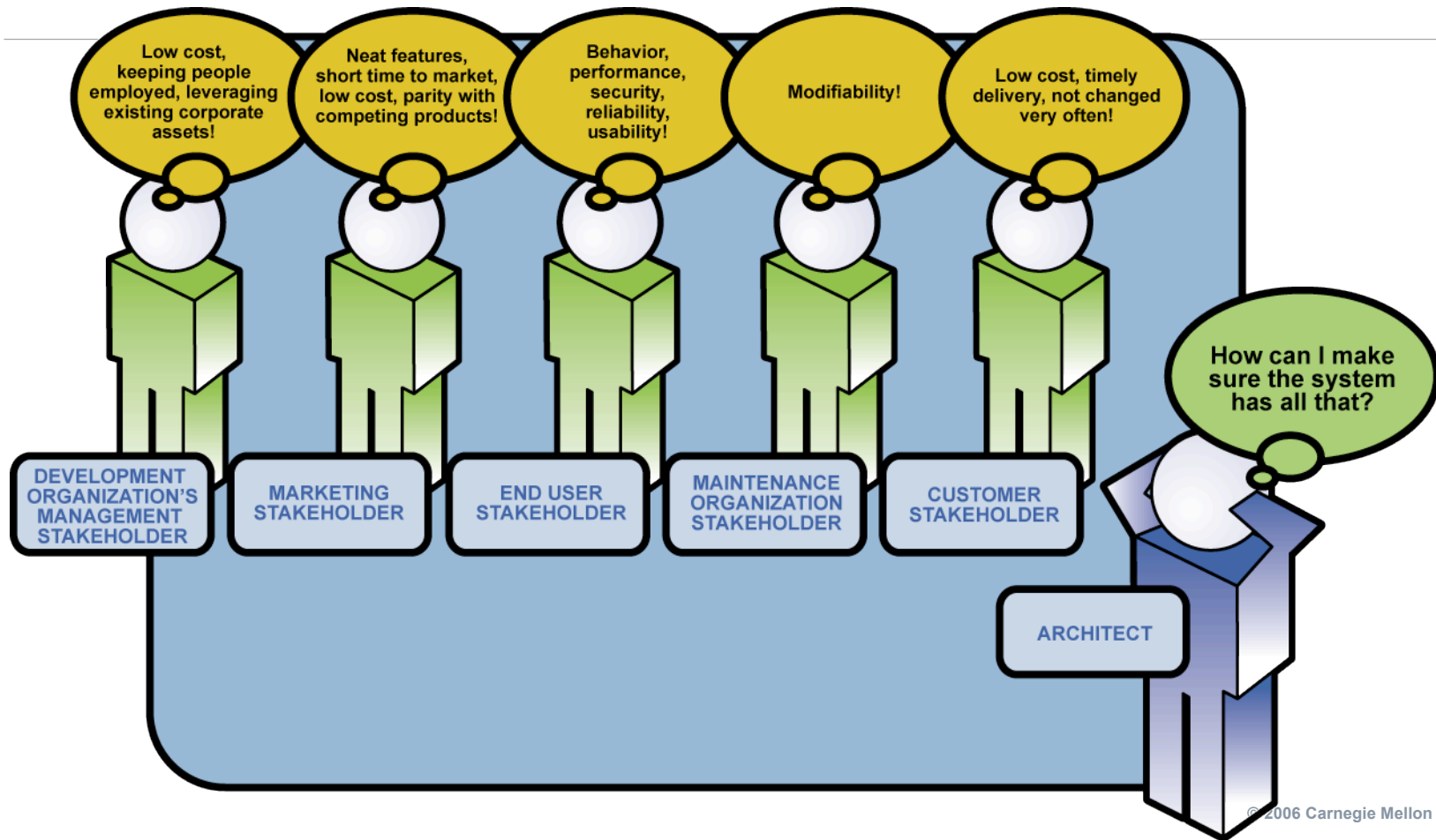
- customers
- users
- developers
- project managers
- marketers
- maintainers

Stakeholders have different concerns that they wish to guarantee and/or optimize.

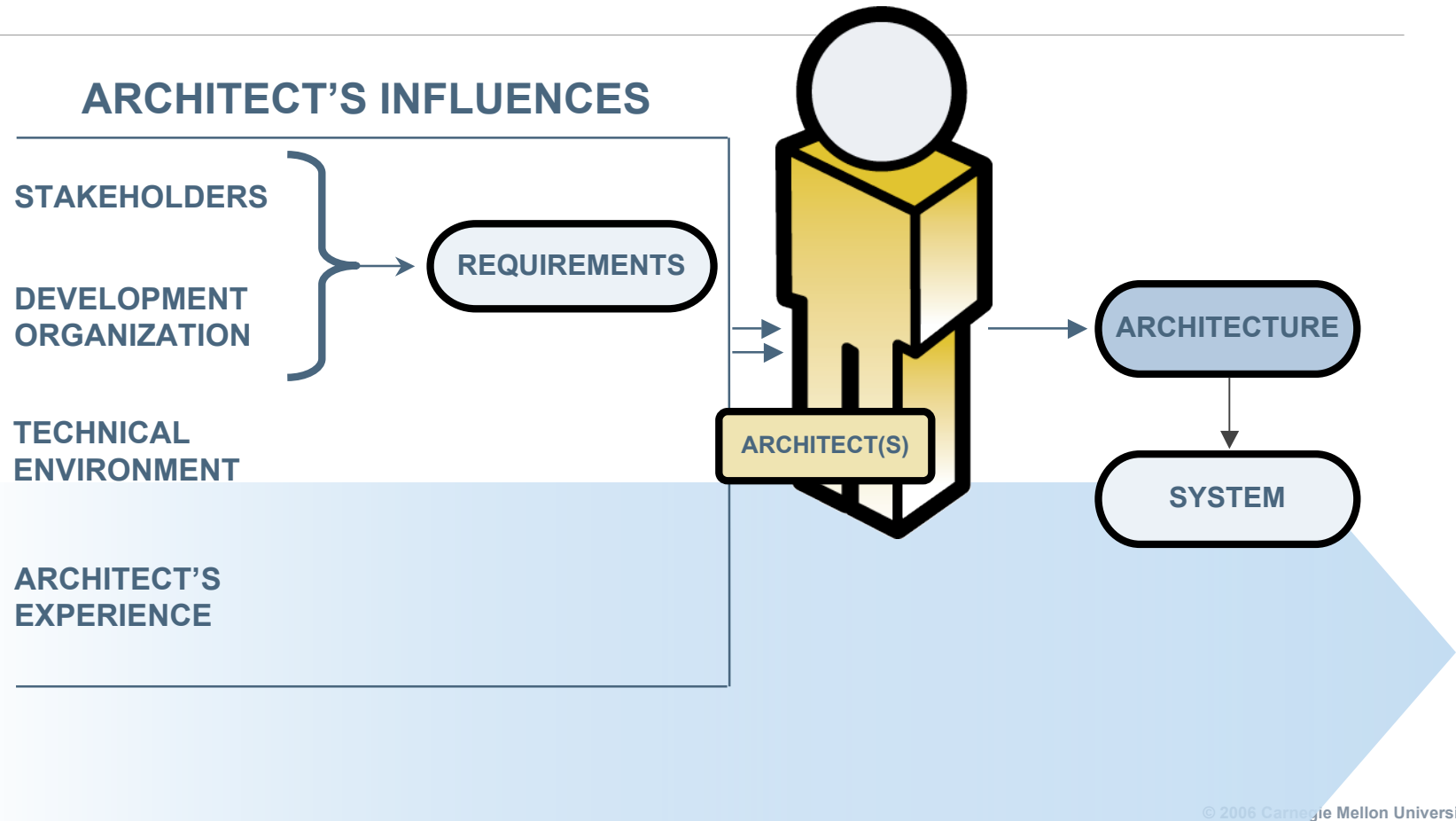
© 2006 Carnegie Mellon University



INFLUENCE OF SYSTEM STAKEHOLDERS – 2

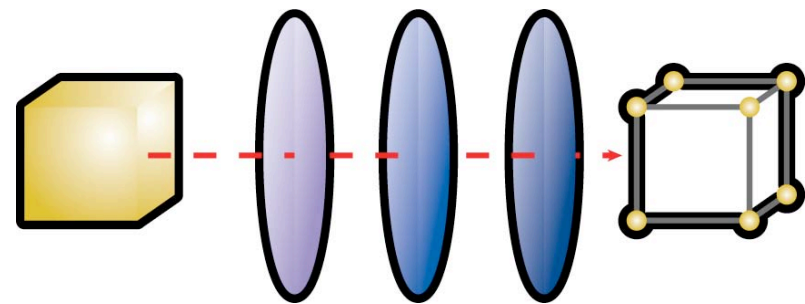


SUMMARY: INFLUENCES ON THE ARCHITECTURE



FACTORS INFLUENCED BY ARCHITECTURES

- Structure of the development organization
- Goals of the development organization
- Customer requirements
- Architect's experience
- Technical environment
- The architecture itself



© 2006 Carnegie Mellon University



A CYCLE OF INFLUENCES

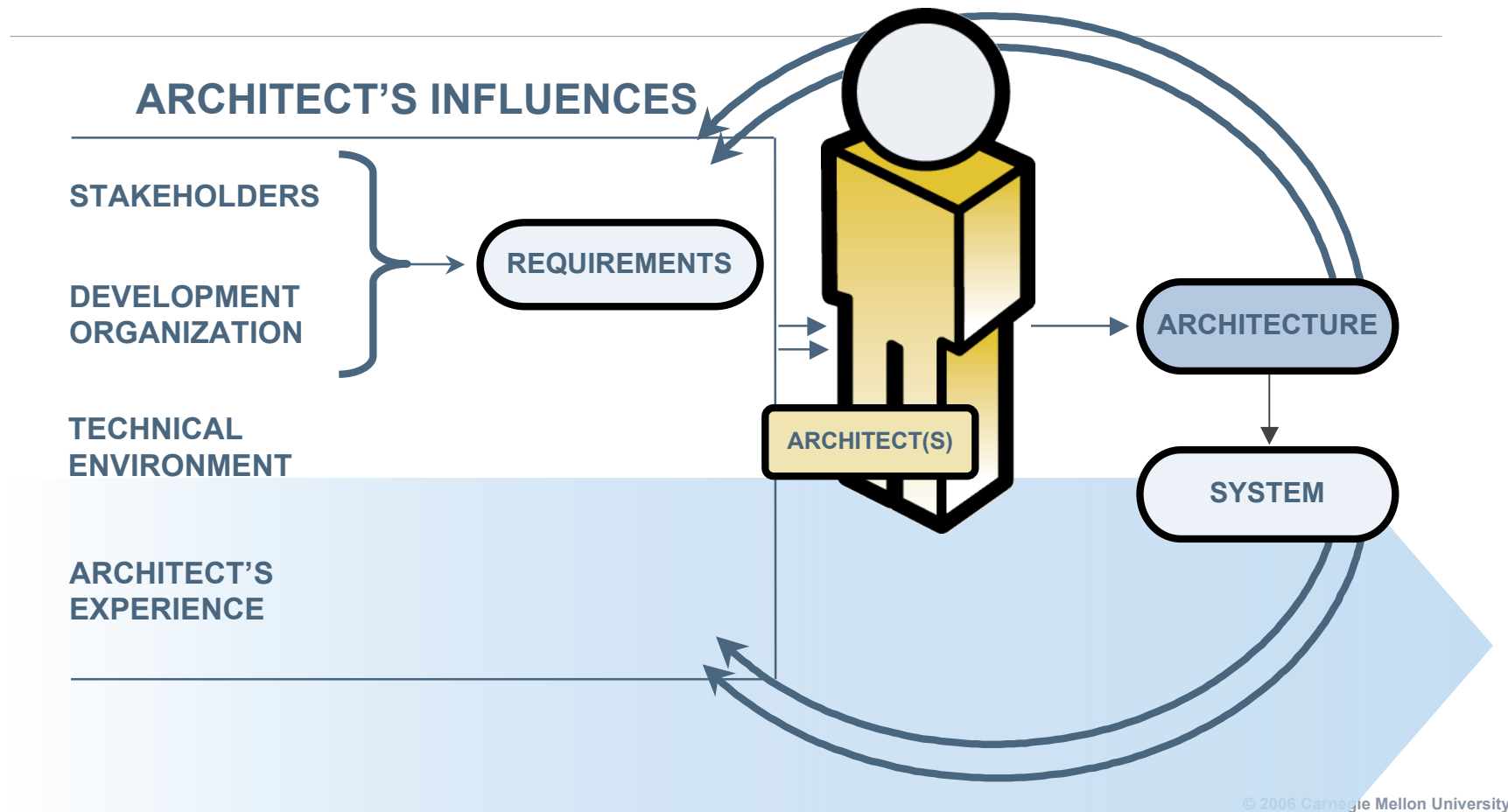
Relationships among business goals, product requirements, architects' experience, architectures and fielded systems form a cycle with feedback loops.

- Influences to and from architectures form a cycle.
- An organization can manage this cycle to its advantage.

© 2006 Carnegie Mellon University



ARCHITECTURE BUSINESS CYCLE (ABC)



SOFTWARE ARCHITECTURE AXIOMS

1. Software architecture is the bridge between business and product goals and a software-intensive system.
2. Quality attribute requirements drive software architecture design.
3. Software architecture drives software development through the life cycle.

© 2006 Carnegie Mellon University



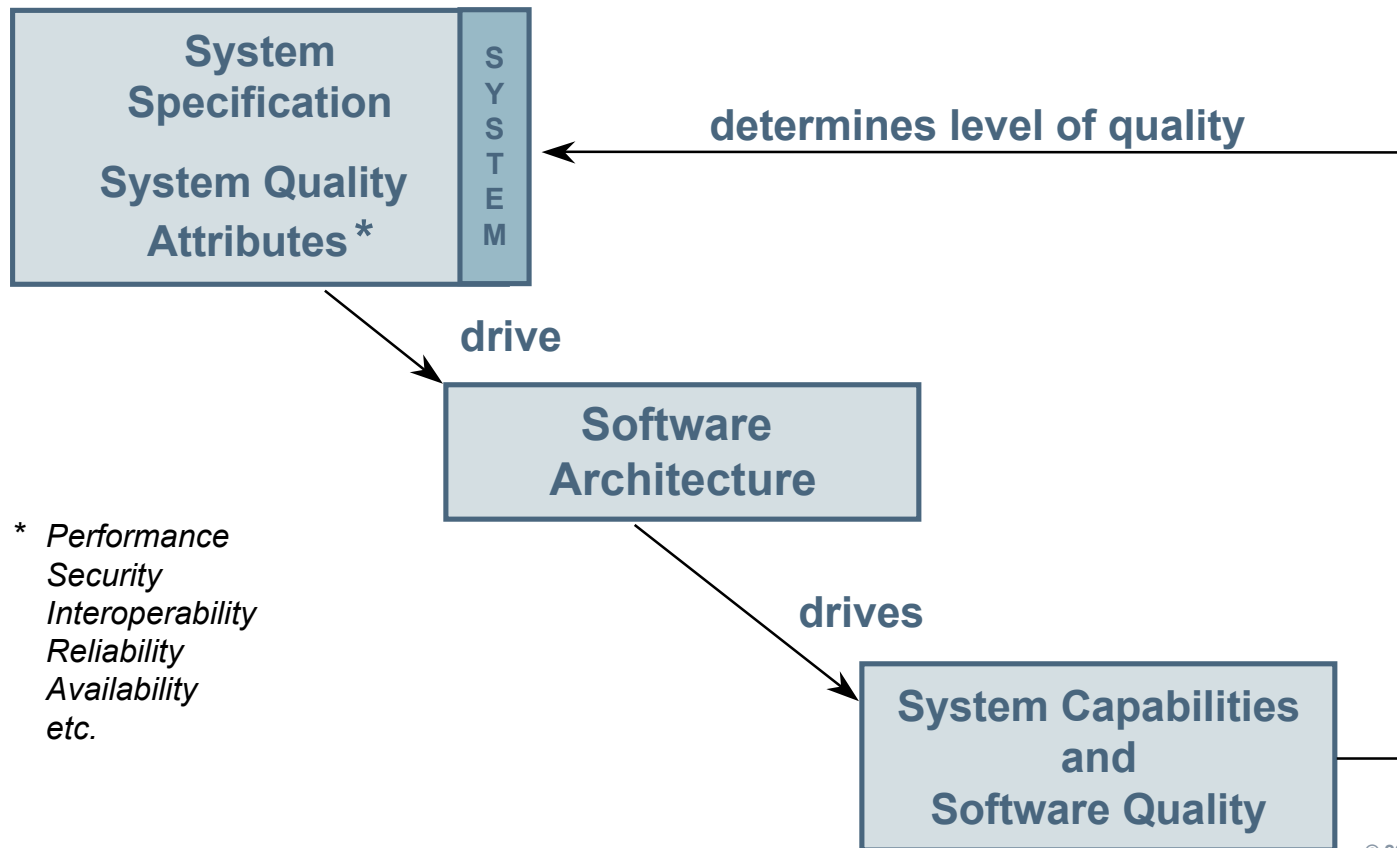
Software Engineering Institute

CarnegieMellon

Let's Teach Architecting High Quality Software

Page 44

SYSTEM QUALITIES AND SOFTWARE ARCHITECTURE



© 2006 Carnegie Mellon University



SOFTWARE ARCHITECTURE COROLLARIES

1. Software architecture is the bridge between business and product goals and a software-intensive system.
2. **Quality attribute requirements drive the design of the software architecture.**
 - Quality attribute requirements stem from business and mission goals.
 - Key quality attributes need to be characterized in a system-specific way.
 - Scenarios are a powerful way to characterize quality attributes and represent stakeholder views.
3. Software architecture drives software development throughout the life cycle.

© 2006 Carnegie Mellon University



SOFTWARE ARCHITECTURE COROLLARIES

1. Software architecture is the bridge between business and product goals and a software-intensive system.
2. Quality attribute requirements drive the software architecture design.
3. **Software architecture drives software development throughout the life cycle.**
 - Software architecture must be central to development activities.
 - These activities must have an explicit focus on quality attributes.
 - These activities must directly involve stakeholders.
 - The architecture must be **descriptive and prescriptive**.

© 2006 Carnegie Mellon University



ARCHITECTURE-CENTRIC DEVELOPMENT ACTIVITIES

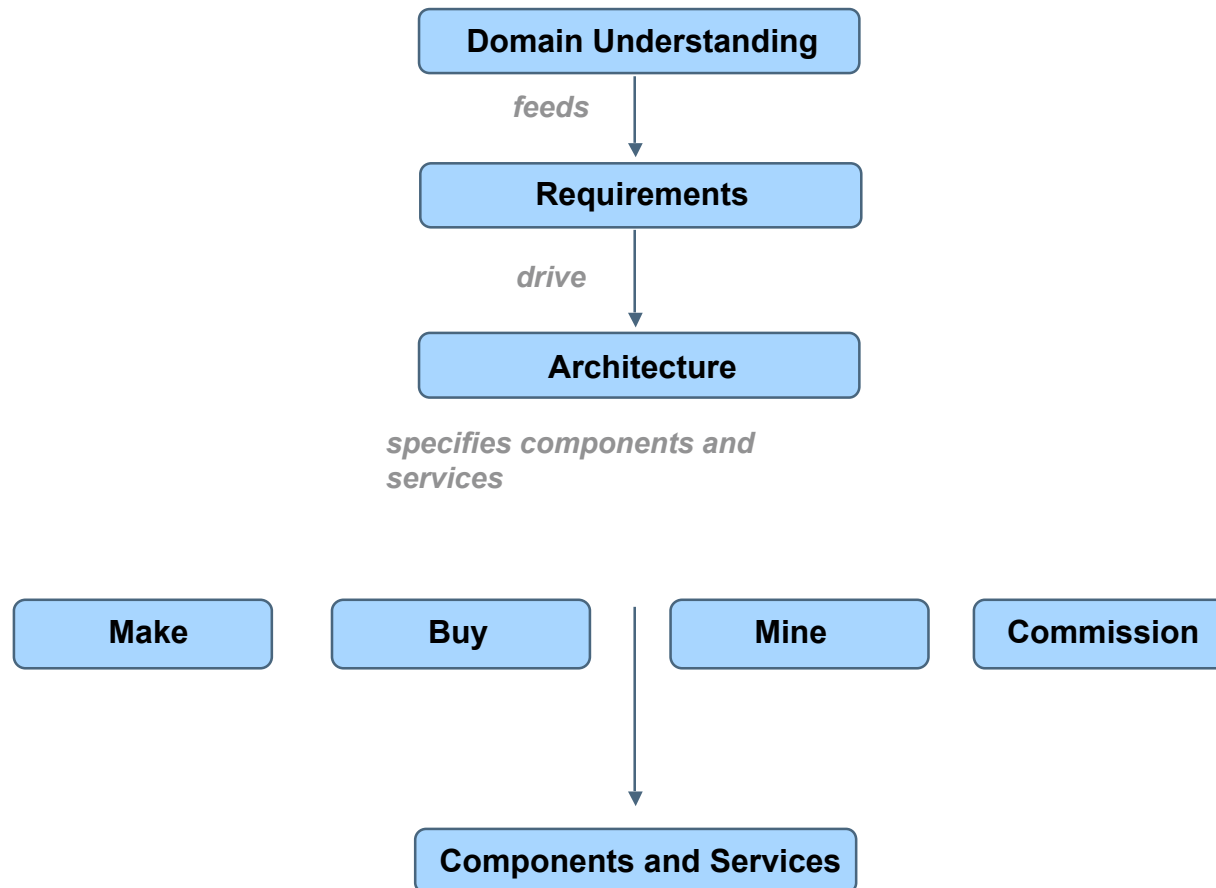
Architecture-centric activities include the following:

- creating the **business case** for the system
- understanding the **requirements**
- **creating and/or selecting** the architecture
- **documenting and communicating** the architecture
- analyzing or evaluating the architecture
- setting up the appropriate **tests and measures** against the architecture
- **implementing** the system based on the architecture
- ensuring that the implementation **conforms** to the architecture

© 2006 Carnegie Mellon University



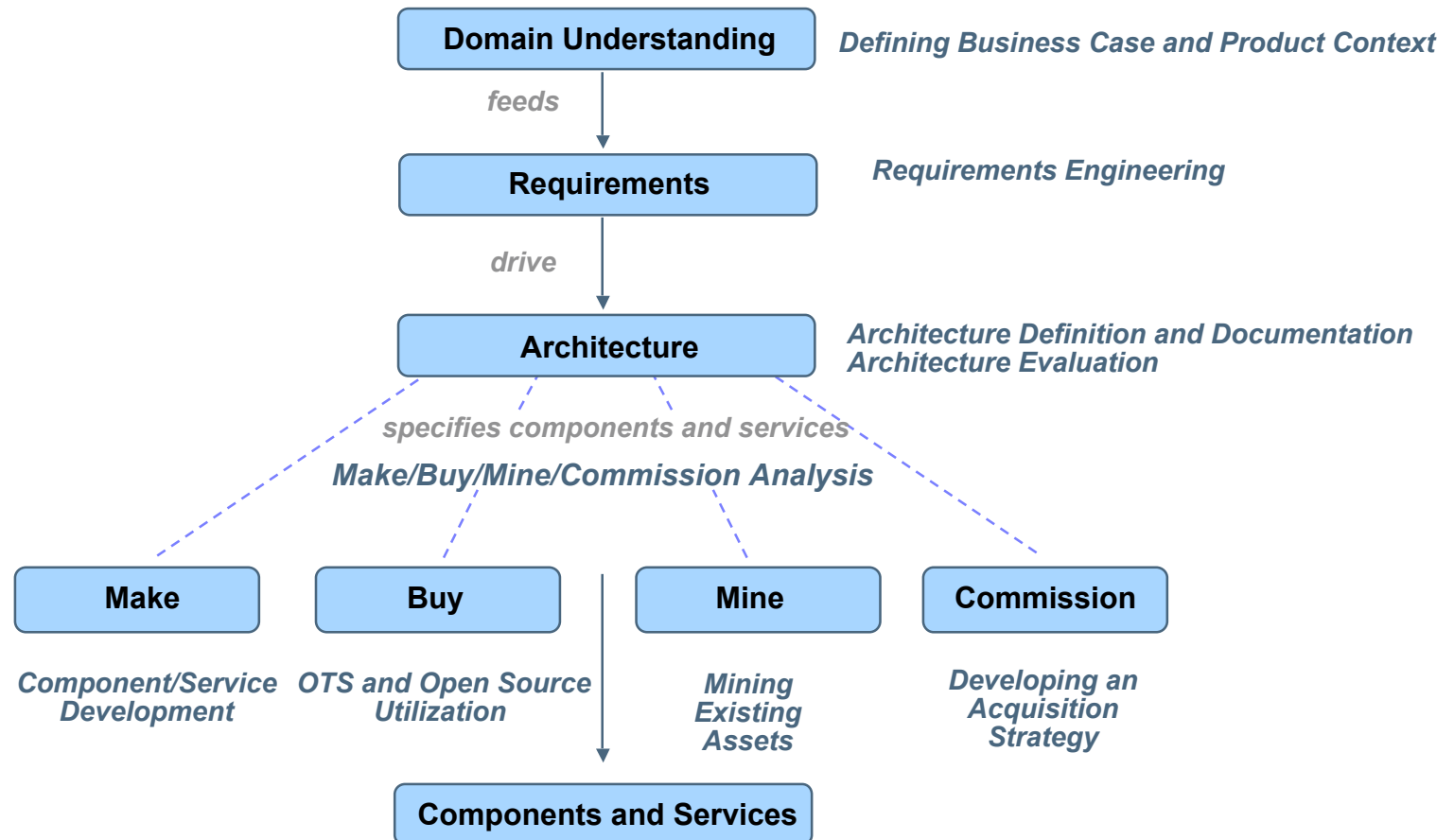
ARCHITECTURE IN THE LIFE CYCLE



© 2006 Carnegie Mellon University



ARCHITECTURE IN THE LIFE CYCLE



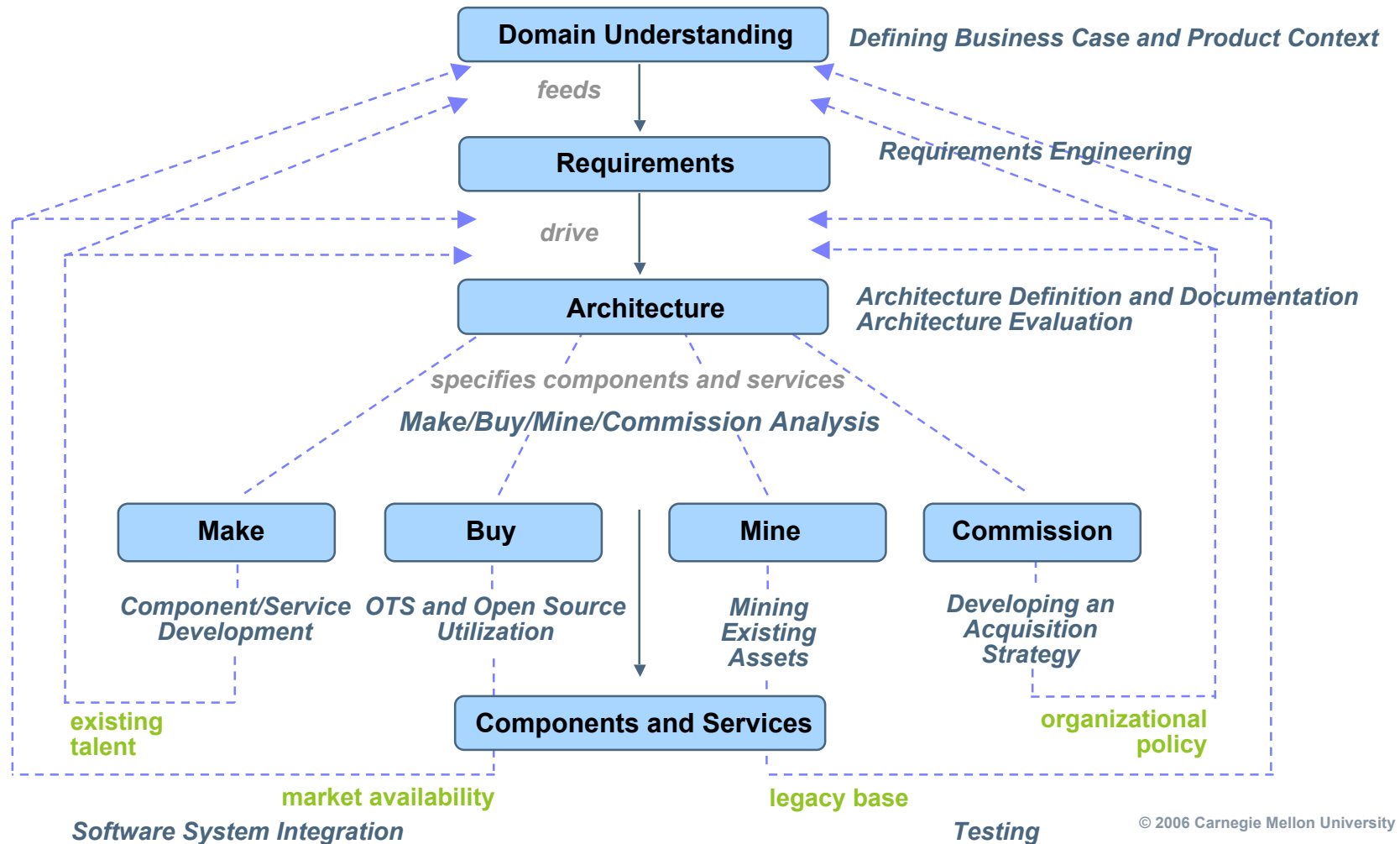
Software System Integration

Testing

© 2006 Carnegie Mellon University



ARCHITECTURE IN THE LIFE CYCLE



STAKEHOLDER INVOLVEMENT

The organizational goals and the system properties required by the business are rarely understood, let alone fully articulated.

Customer quality attribute requirements are seldom documented, which results in

- goals not being achieved
- inevitable conflict between different stakeholders

Architects must identify and actively engage stakeholders in order to

- understand real constraints of the system
- manage the stakeholders' expectations
- negotiate the system's priorities
- make tradeoffs

© 2006 Carnegie Mellon University



WHAT MAKES A GOOD ARCHITECTURE?

There is no such thing as an inherently good or bad architecture.
Architectures are more or less fit for some stated purpose.

The “goodness” of an architecture can be determined with respect to
business and product goals

- Assume that two systems are functionally identical. One system can only be “better” if its architecture promotes qualities that are required to meet business goals and/or product goals.

© 2006 Carnegie Mellon University



IMPEDIMENTS TO ACHIEVING ARCHITECTURAL SUCCESS

Lack of

- adequate architectural talent and/or experience
- time spent on architectural design and analysis
- architecture-centric acquisition practices

Failure to

- identify the key quality attributes, characterize them, and design for them
- properly document and communicate the architecture
- evaluate the architecture in a qualitative way
- understand that standards are not a substitute for a software architecture
- ensure that the architecture directs the implementation
- evolve the architecture and maintain documentation that is current
- understand that a software architecture does not come free with COTS or services

© 2006 Carnegie Mellon University



CHALLENGES FOR THE ARCHITECT

- What are the driving quality attributes for your system?
- What precisely do these quality attributes such as modifiability, security, performance, and reliability mean?
- How do you architect to ensure the system will have its desired qualities?
- How do you document a software architecture?
- How do you know if software architecture for a system is suitable without having to build the system first?
- Can you recover an architecture from an existing system?

© 2006 Carnegie Mellon University



SOME SEI TECHNIQUES AND METHODS

- creating the **business case** for the system
- understanding the **requirements**
 - *Quality Attribute Workshop (QAW)*
- **creating and/or selecting** the architecture
 - *Attribute-Driven Design (ADD) and ArchE*
- **documenting and communicating** the architecture
 - *Views and Beyond Approach*
- **analyzing or evaluating** the architecture
 - *Architecture Tradeoff Analysis Method (ATAM)*
 - *Cost Benefit Analysis Method (CBAM)*
- **implementing** the system based on the architecture
- ensuring that the implementation **conforms** to the architecture
 - *ARMIN (and DiscoTect)*

© 2006 Carnegie Mellon University



SEI METHODS AND QUALITY ATTRIBUTES

QAW

ADD

ArchE

Views and Beyond

ATAM

CBAM

ARMIN

(DiscoTect)



- are explicitly focused on quality attributes
- directly link to business and mission goals
- explicitly involve system stakeholders
- are grounded in state-of-the-art quality attribute models and reasoning frameworks
- are documented for practitioner consumption
- are applicable to real-world challenges and systems

© 2006 Carnegie Mellon University



TRENDS IN SOFTWARE ARCHITECTURE - 1

Organizations big and small are recognizing the importance of software architecture. For example,

- Microsoft
 - Regional Architecture Forums
 - Architect's Council
 - Architect Certification
- Raytheon
 - Architecture Center of Excellence
 - mandatory architecture classes and methods
- IBM
 - Grady Booch writing the online Architect's Handbook
- Automotive domain
 - Siemens, Bosch, and Delphi all have architecture initiatives
- US Army
 - Army Software Architecture Initiative

© 2006 Carnegie Mellon University



TRENDS IN SOFTWARE ARCHITECTURE - 2

Books, courses, certificate programs, conferences, workshops on software architecture abound.

New technologies (MDA, SOA, aspects) change the incidentals but the fundamentals of software architecture and quality attributes are enduring.

© 2006 Carnegie Mellon University



Let's Teach Architecting High Quality Software

Teach

© 2006 Carnegie Mellon University



Software Engineering Institute

CarnegieMellon

Let's Teach Architecting High Quality Software

Page 60

SOFTWARE ENGINEERING PERSPECTIVES - 1

PROCESS	PRODUCT

© 2006 Carnegie Mellon University



Software Engineering Institute

CarnegieMellon

Let's Teach Architecting High Quality Software

Page 61

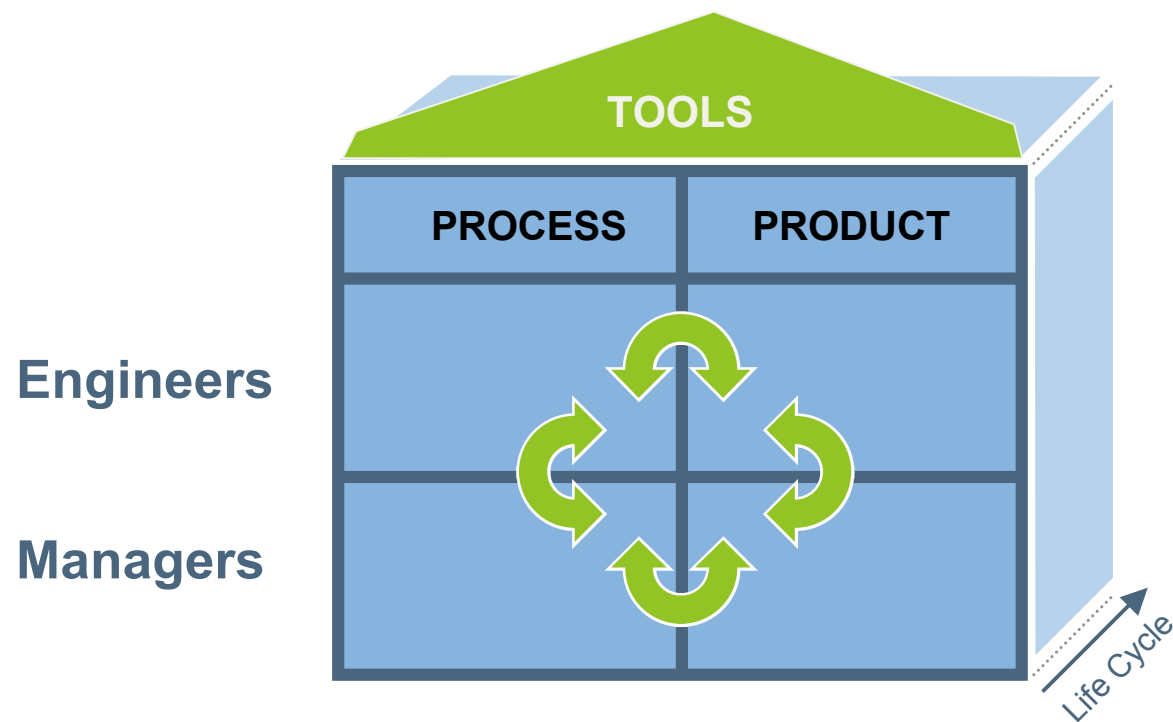
SOFTWARE ENGINEERING PERSPECTIVES - 2

	PROCESS	PRODUCT
Engineers		
Managers		

© 2006 Carnegie Mellon University



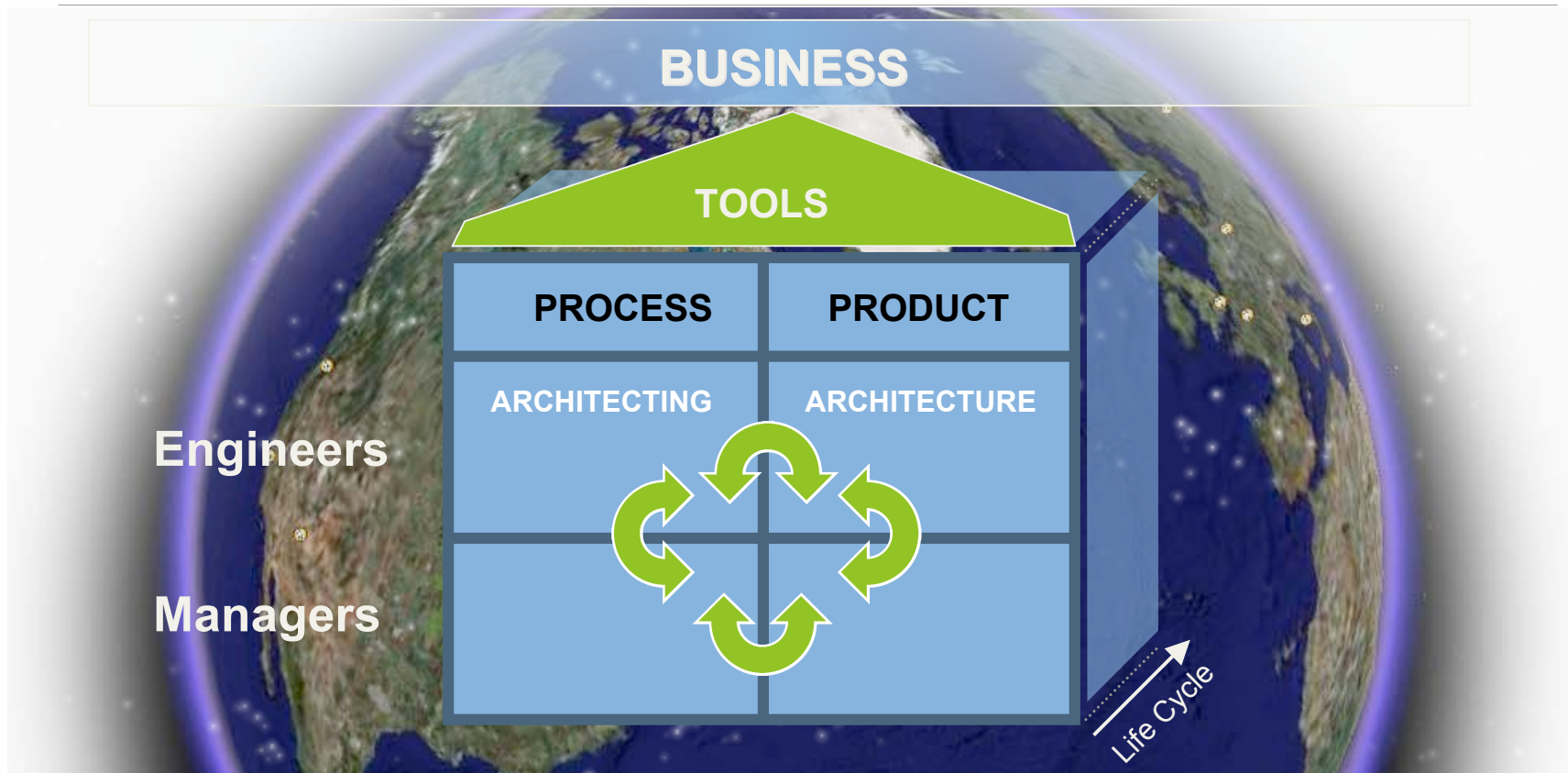
SOFTWARE ENGINEERING PERSPECTIVES - 3



© 2006 Carnegie Mellon University



SOFTWARE ARCHITECTURE



BOTH INFLUENCES AND IS INFLUENCED BY ALL PERSPECTIVES



SOFTWARE ENGINEERING EDUCATION

What do we teach and how does it address this model?

The norm

Product perspective: OOA + OOD + OOP + ...

Process perspective: agile methods, XP, TSP + ...

These are a good starting perspective but software engineering students need more

- Business context
- Quality attributes
- Software architecture

© 2006 Carnegie Mellon University



ARCHITECTURE PRINCIPLES TO TAKE AWAY

Software architecture is important because it

- provides a communication vehicle among stakeholders
- is the result of the earliest design decisions
- is a transferable, reusable abstraction of a system

The degree to which a system meets its quality attribute requirements is dependent on architectural decisions.

Every software-intensive system has a software architecture.

Just having an architecture is different from having an architecture that is known to everyone, much less one that is fit for the system's intended purpose.

An architecture-centric approach is critical to achieving and implementing an appropriate architecture.

High quality software requires architecture practices.

© 2006 Carnegie Mellon University



THE ROLE OF EDUCATORS



© 2006 Carnegie Mellon University



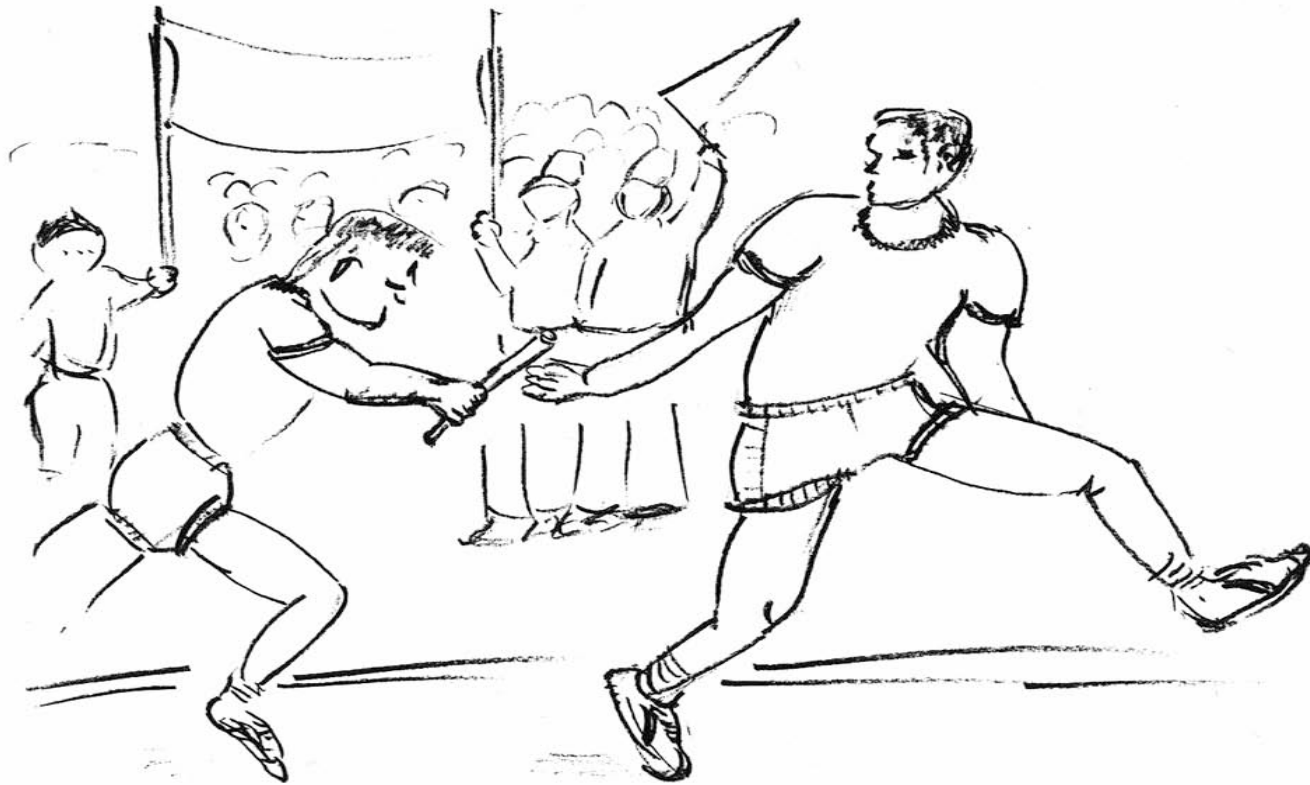
Software Engineering Institute

CarnegieMellon

Let's Teach Architecting High Quality Software

Page 67

THE WELL EDUCATED GRADUATE



© 2006 Carnegie Mellon University



Software Engineering Institute

CarnegieMellon

Let's Teach Architecting High Quality Software

Page 68

LET'S TEACH ARCHITECTING HIGH QUALITY SOFTWARE

It's time that all software engineering students know the principles of software architecture and how to use effective software architecture practices.

Every facet of our society depends on software.

To ensure high quality software we need to teach our students to architect high quality software.

© 2006 Carnegie Mellon University



THANKS TO THE SEI SOFTWARE ARCHITECTURE TEAM

Felix Bachmann, Len Bass,
Joe Batman, John Bergey,
Phil Bianco, Paul Clements,
James Ivers, Larry Jones,
Rick Kazman, Mark Klein,
Reed Little, Paulo Merson,
Robert Nord, William
O'Brien, Ipek Ozkaya, Rob
Wojcik, Bill Wood



© 2006 Carnegie Mellon University



THANKS TO MY DEAR FRIEND AND COLLEAGUE



Coach



I dedicate this keynote to Jim Tomayko. Jim was a leading contributor to software engineering education and a role model to all educators.

© 2006 Carnegie Mellon University



Software Engineering Institute

CarnegieMellon

Let's Teach Architecting High Quality Software

Page 71

THANK YOU!

It has been my honor and pleasure to spend this time with you.

Linda Northrop

Director

Product Line Systems Program

Telephone: 412-268-7638

Email: lmn@sei.cmu.edu

For more information:

http://www.sei.cmu.edu/architecture/sat_init.html

© 2006 Carnegie Mellon University



Software Engineering Institute

CarnegieMellon

Let's Teach Architecting High Quality Software

Page 72

REFERENCES

Evaluating Software Architectures: Methods and Case Studies

Clements, P.; Kazman, R.; & Klein, M. Reading, MA:
Addison- Wesley, 2002.

Software Architecture in Practice, Second Edition

Bass, L.; Clements, P.; & Kazman, R. Reading, MA:
Addison-Wesley, 2003.

Documenting Software Architectures: Views and Beyond

Clements, P.; Bachmann, F.; Bass, L.; Garlan, D.; Ivers, J.; Little, R.;
Nord, R.; & Stafford, J. Reading, MA:
Addison-Wesley, 2002.

Software Product Lines: Practices and Patterns

Clements, P.; Northrop, L. Reading, MA: Addison-Wesley, 2001.

© 2006 Carnegie Mellon University

